NPS-MA-91-009

NAVAL POSTGRADUATE SCHOOL
Monterey, California
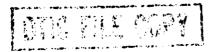
AD-A235 489

DTIC
ELECTE
MAY 0 2 1991
S B D

SOFTWARE FOR THE PARALLEL SOLUTION
OF SYSTEMS OF ORDINARY
DIFFERENTIAL EQUATIONS

Levi Lustman

Beny Neta

February 1991

Prepared for: Naval Postgraduate School
Monterey, CA 93943

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS	
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-MA-91-009		5 MONITORING ORGANIZATION REPORT NUMBER(S) NPS-MA-91-009	
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (If applicable) MA	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School	8b OFFICE SYMBOL (If applicable) MA	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER O&MN Direct Funding	
8c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		10 SOURCE OF FUNDING NUMBERS	

			10 SOURCE OF FUNDING NUMBERS			
PROGRAM ELEMENT NO	PROJECT NO	TASK NO.	WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

Software for the Parallel Solution of Systems of Ordinary Differential Equations

12 PERSONAL AUTHOR(S)
Levi Lustman and Beny Neta

13a TYPE OF REPORT Technical Report	13b TIME COVERED FROM 1/2/91 TO 2/25/91	14 DATE OF REPORT (Year, Month, Day) 28 February 1991	15 PAGE COUNT 28

16 SUPPLEMENTARY NOTATION

17	COSATI CODES		18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	ordinary differential equations, parallel processing, hypercube

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This report contains software for the solution of systems of ordinary differential
equations on an INTEL iPSC/2 hypercube. A diskette is available upon request
from the second author.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Levi Lustman and Beny Neta	22b TELEPHONE (Include Area Code) (408) 646-2206	22c OFFICE SYMBOL MA/Nd

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Software for the Parallel Solution of Systems of Ordinary Differential Equations

L. Lustman
B. Neta

Naval Postgraduate School
Department of Mathematics
Code MA
Monterey, CA 93943

Abstract

This report contains software for the solution of systems of ordinary differential equations on an INTEL iPSC/2 hypercube. A diskette is available upon request from the second author.

1.   Introduction

In this report we supply software for the numerical solution of systems of ordinary differential equations (ODEs) on an INTEL iPSC/2 hypercube. The first program can only be used to solve *linear* initial or boundary value systems of ODEs and based on an algorithm developed by Katti and Neta (1989) and improved by Lustman *et al* (1990). The second program is based on polynomial extrapolation and Gragg's scheme and is useful for nonlinear ODEs as well. This algorithm is described in Lustman, Neta and Gragg (1991).

2. Linear Systems

In this section we give the software for the solution of *linear* systems of ODEs:

$$y'(x) = Ay(x) + g(x), \ a < x < b$$
$$y(a) = y_a$$

(1)

The algorithm used was developed by Katti and Neta (1989) and improved by Lustman *et al* (1990). The host and node program are given. The subroutines sa, sf and putex give the matrix A, the right hand side of (1) and the exact solution (for debugging purposes) respectively. An example of input and output corresponding to these subroutines are attached.

```
c
c
c                         HOST
c solving initial value problems by multiple shooting
c on INTEL iPSC/2 hypercube having 8 (maxnp) processors
c
c                  see  Lustman, Neta & Katti
c
c change everywhere, in both node and host programs,
c                 ndim=3
c to whatever value is appropriate.
c
            program mshivph
            integer intype,inlen,outype,outlen
            integer ymtype,ymlength
            integer n ,np,ndim,nin,nout, m , mnp
            integer allnodes,hostpid,nodepid
            parameter (nmax=100)
            parameter (ndim=3,nout=1)
           parameter(maxnp=8)
            parameter (nin=nmax*maxnp+ndim+10)
            parameter (intype=10,outype=20,inlen=4*nin
     #  ,ymtype=30,ymlength=ndim*(ndim+1)*4
     #  ,outlen=4*nout,allnodes= -1
     #  ,hostpid=8,nodepid=14)
           common/cin/n,ndimc,ninc,noutc
     #,m,mp,h,left,right,g,x
            real g (ndim) , x (0:nmax*maxnp) , vin (1)
            real vout (nout) , left , right
           equivalence
     # (n,vin(1)),(ndimc,vin(2)),(ninc,vin(3))
     #,(noutc,vin(4)),(m,vin(5)),(mp,vin(6))
     #,(h,vin(7)),(left,vin(8)),(right,vin(9))
     #,(g(1),vin(10))
     #,(x(0),vin(10+ndim))
           ndimc=ndim
           ninc=nin
           noutc=nout
           call getcube('shoot',' ',' ',1)
           call setpid(hostpid)
            print*,' got the maximal cube,',numnodes(),' nodes'
           call load('node',allnodes,nodepid)
           print*,' after load'
           print*,' enter ',ndim,' initial values g'
           read*,(g(i),i=1,ndim)
           print*,' enter endpoints of interval'
           read*,left,right
           print*,'solve for ',left,' <x< ',right
     ,,' initially=',(g(i),i=1,ndim)
           print*,' enter number of points in interval, for each processor'
           read*,m
           print*,m,' points for each processor'
           np=numnodes()
           mnp=m*np
           h=(right-left)/mnp
```

4

```
         do 400 i=0,mnp
400      x(i)=left+(i)*h
         call csend(intype,vin,inlen,allnodes,nodepid)
411      continue
         call waitall(allnodes,nodepid)
         call relcube('shoot')
         stop
         end
```

```
c'
c                          NODE
c solving initial value problems by multiple shooting
c on INTEL iPSC/2 hypercube having 8 (maxnp) processors
c
c                    see  Lustman, Neta & Katti
c
c Change everywhere, in both node and host programs,
c                    ndim=3
c to whatever value is appropriate.
c
c The subroutines        sa (computing the matrix A) and
c                         sf (the right hand side)
c                         putex ( the exact solution, needed for debugging
c
c must be supplied for each application. (Examples are given in the code
c
             program MSHIVPN
             integer intype,inlen,outype,outlen
             integer ymtype,ymlen,ymdim,cubdimax
             integer n ,np,ndim,nin,nout, m , mnp ,ready
             integer allnodes,hostpid,nodepid
            integer tend,tbeg
             parameter (nmax=100)
             parameter (ndim=3,nout=1)
            parameter(maxnp=8)
             parameter (nin=nmax*maxnp+ndim+10)
             parameter (intype=10,outype=20,inlen=4*nin
     # ,cubdimax=3,ymtype=300,ymdim=ndim*(ndim+1) )
             parameter (ymlen=4+ymdim*4
     # ,outlen=4*nout,allnodes= -1
     # ,hostpid=8,nodepid=14)
             common/cin/n,ndimc,ninc,noutc
     #,m,mp,h,left,right,g,x
             real g (ndim) , x (0:nmax*maxnp) , vin (nin)
             real vym(0:ymdim,0:cubdimax)
             real vym0(0:ymdim)
             real vout (nout) , left , right
             equivalence
     # (n,vin(1)),(ndimc,vin(2)),(ninc,vin(3))
     #,(noutc,vin(4)),(m,vin(5)),(mp,vin(6))
     #,(h,vin(7)),(left,vin(8)),(right,vin(9))
     #,(g(1),vin(10))
     #,(x(0),vin(10+ndim))
             dimension phiex(ndim),phi(ndim),ytilde(ndim)
             dimension er(ndim)
             dimension ucphi(ndim,ndim),binv(ndim,ndim)
             real a(ndim,ndim),b(ndim,ndim)
             dimension ynit(ndim),partic(ndim)
             call crecv(intype,vin,inlen)
             me=mynode()
             numno=numnodes()
             jl=me*m
             jh=jl+m
c
```

```
c`initialization
c
        call init(ndim,ucphi,ytilde)
        xme=jh*h+left
cdebug  call putex(xme,phiex,g)
        do 100 j=jl,jh-1
        xx=x(j)+0.5*h
c
c get A
c
        call sa(ndim,xx,a)
c
c get B=I - h/2 A
c
        call sb(h,ndim,a,b)
c
c evaluate B inverse
c
        call sbinv(b,binv,ndim)
c
c evaluate D = Binv *(I + h/2 A)
c
        call sa(binv,h,ndim,a,b)
c
c multiply ucphi*B
c
        call smult(ucphi,b,ndim)
c
c get right hand side
c
        call sf(ndim,xx,f)
c
c get phi
c
        call sphi(b,ytilde,h,binv,f,ndim,phi)
c
c copy phi to ytilde
c
        if(j.lt.jh-1) then
        call scopy(phi,ytilde,ndim)
        endif
 100    continue
c
c the following starts with initial conditions
c
        if(me.eq.0) call sma(ucphi,g,phi,ndim)
c
c here the process of recursive doubling
c
        jq=me+1
        iq=1
1132    continue
c
c send to  some node after me
c
```

7

```fortran
             if(jq+iq.le.numno) then
c
c make a list of data to send in the buffer vym0
c
        call enlist(me,phi,ucphi,vym0,ndim)
        call csend(ymtype+me,vym0,ymlen,iq+me,nodepid)
        endif
c
c   y1j = bj =phi j
c   m1j = phi j
c
 1133   continue
        if(me.ge.iq) then
c
c       me  requires data from  me-iq
c
c
        call crecv (ymtype+me-iq,vym0,ymlen)
        do 58 i=1,ndim+ndim*ndim
58      vym(i,1)=vym0(i)
c
c y1 =y1 + M * y0
c
        call defy(ndim,phi,ucphi,vym(1,1))
c
c M = M * M0
c
        call defm(ndim,ucphi,vym(ndim+1,1))
        endif
        iq=2*iq
        if(iq.lt.numno) goto 1132
c
c end of processing
c
c       iunit=10+me
cdebug  do 1001 i=1,ndim
cdebug 1001     er(i)=abs(phi(i)-phiex(i))
        print1000,xme,phi
 1000   format('x=',f6.2,' phi=',3f6.2)
cdebug  print1001,er
cdebug 1001     format(8x,' err=',3f6.2)
        stop
        end
c
c makes a list of values to send in the buffer v
c
        subroutine enlist(me,phi,ucphi,v,n)
        dimension v(0:1),phi(n),ucphi(n,n)
        v(0)=me
        l=1
        do 1 i=1,n
        v(l)=phi(i)
        l=l+1
 1      continue
        do 2 j=1,n
```

8

```
              do 2 i=1,n
              v(l)=ucphi(i,j)
              l=l+1
  2           continue
              return
              end
c
c computes B= I - h/2 A
c
              subroutine sb(h,ndim,a,b)
c evaluate b=i-h/2*a
              real a(ndim,ndim),b(ndim,ndim)
              do 10 i=1,ndim
              do 10 j=1,ndim
              r=0
              if(i.eq.j) r=1
              b(i,j)=r-0.5*h*a(i,j)
  10          continue
              return
              end
c
c computes D= Binv * ( I + h/2 A )
c
              subroutine sd(binv,h,ndim,a,b)
              real a(ndim,ndim),b(ndim,ndim),binv(ndim,ndim)
              do 10 i=1,ndim
              do 10 j=1,ndim
              b(i,j)=0
              do 10 k=1,ndim
              r=0
              if(k.eq.j) r=1
              b(i,j)=b(i,j)+binv(i,k)*(r+0.5*h*a(k,j))
  10          continue
              return
              end
c
c evaluate b*ucphi into ucphi
c
              subroutine smult(ucphi,b,idim)
               parameter (ndim=3)
              real ucphi(idim,idim),b(idim,idim)
              real temp(ndim)
              do 100 j=1,idim
              do 10 i=1,idim
              temp(i)=0
              do 10 k=1,idim
              temp(i)=temp(i)+b(i,k)*ucphi(k,j)
  10          continue
              do 20 k=1,idim
  20          ucphi(k,j)=temp(k)
  100         continue
              return
              end
c
c evaluate d*ytilde + h*binv*f
```

```fortran
c
      subroutine sphi(b,ytilde,h,binv,f,ndim,phi)
      real b(ndim,ndim),ytilde(ndim),binv(ndim,ndim)
      real f(ndim),phi(ndim)
      do 10 i=1,ndim
      phi(i)=0
      do 10j=1,ndim
10    phi(i)=phi(i)+b(i,j)*ytilde(j)+h*binv(i,j)*f(j)
      return
      end
c
c moves phi to ytilde
c
      subroutine scopy(phi,ytilde,ndim)
      real ytilde(ndim),phi(ndim)
      do 10 i=1,ndim
10    ytilde(i)=phi(i)
      return
      end
c
c evaluate ucphi*g +phi and put into phi
c
      subroutine sma(ucphi,g,phi,ndim)
      real phi(ndim),ucphi(ndim,ndim),g(ndim)
      do 10 i=1,ndim
      do 10 j=1,ndim
10    phi(i)=phi(i)+ucphi(i,j)*g(j)
      return
      end
c
c initialize ucphi and ytilde
c
      subroutine init(ndim,ucphi,ytilde)
      real ytilde(ndim),ucphi(ndim,ndim)
      do 10 i=1,ndim
      ytilde(i)=0
      do 20 j=1,ndim
      ucphi(i,j)=0
20    continue
      ucphi(i,i)=1
10    continue
      return
      end
c
c inverts b into binv . b is destroyed
c                       ==================
c
      subroutine sbinv(b,binv,ndim)
      real b(ndim,ndim),binv(ndim,ndim)
      do 20 i=1,ndim
      do 10 j=1,ndim
10    binv(i,j)=0
20    binv(i,i)=1
      do 2 j=1,ndim
      z=1/b(j,j)
```

```fortran
              do 30 k=1,ndim
              b(j,k)=z*b(j,k)
              binv(j,k)=z*binv(j,k)
30            continue
              do 1 i=1,ndim
              if(i.eq.j) goto 1
              z=b(i,j)
              do 3 k=1,ndim
              b(i,k)=b(i,k)-z*b(j,k)
              binv(i,k)=binv(i,k)-z*binv(j,k)
3             continue
1             continue
2             continue
              return
              end
c
c evaluates Y1=Y1+M*Y0
c
              subroutine defy(ndim,y1,em,y0)
              dimension y1(ndim),em(ndim,ndim),y0(ndim)
              do 1 i=1,ndim
              do 1 j=1,ndim
              y1(i)=y1(i)+em(i,j)*y0(j)
1             continue
              return
              end
c
c evaluates M=M*M0
c
              subroutine defm(ijmax,em,em0)
               parameter (ndim=3)
              dimension row(ndim)
              dimension em(ijmax,ijmax),em0(ijmax,ijmax)
              do 1 i=1,ijmax
              do 3 j=1,ijmax
              row(j)=em(i,j)
3             continue
              do 1 j=1,ijmax
              s=0
              do 2 k=1,ijmax
              s=s+row(k)*em0(k,j)
2             continue
              em(i,j)=s
1             continue
              return
              end
cdebugc
cdebugc given x, and initial values g, computes v=exact(x)
cdebugc
cdebug    subroutine putex(x,v,g)
cdebug    parameter (ndim=3)
cdebug    parameter(e=2.718281828,ei=1./e)
cdebug    dimension v(ndim),g(ndim)
cdebug    dimension v(3)
cdebug    real 10g
```

11

```
cdebug   ex=exp(x)
cdebug   10g=alog(x)
cdebug   a=(g(1)-1)*ei
cdebug   b=(g(2)-e)*ei
cdebug   c=(g(3)-ei)*ei
cdebug   v(1)=ex*(a+10g*(b+c/2*10g))+1
cdebug   v(2)=ex*(b+c*10g)+ex
cdebug   v(3)=ex*c+1/ex
cdebug   return
cdebug   end
c
c evaluate right hand side f(x)
c
         subroutine sf(idim,x,f)
         parameter (ndim=3)
         real x, f(idim)
         ex=exp(x)
         f(1)=-1-ex/x
         f(2)=-1/x/ex
         f(3)=-2/ex
         return
         end
c
c evaluate the matrix A(x)
c
         subroutine sa(ndim,x,a)
         real a(ndim,ndim),x
         do 10 i=1,ndim
         do 10 j=1,ndim
         a(i,j)=0
  10     continue
         a(1,1)=1
         a(2,2)=1
         a(3,3)=1
         a(1,2)=1/x
         a(2,3)=1/x
         return
         end
```

12

```
# This file is used to compile and link the host.f, node.f
#
# The command "make all" causes compilation and linking.


all :   host node


host:   host.o
        f77 -o host host.o -host


node:   node.f
        f77 -o node node.f -node
```

```
**********************************************
            example of an input file
            for the subroutine sa, sf, putex
            currently in node.f
**********************************************
0,0,0   initial values
1,2         endpoints
5       subintervals for each processor



**************************************************
            example of output file for the above
**************************************************
  got the maximal cube,              8 nodes
  after load
  enter               3 initial values g
  enter endpoints of interval
 solve for      1.000000      <x<      2.000000
initially=  0.0000000E+00   0.0000000E+00   0.0000000E+00
  enter number of points in interval, for each processor
            5 points for each processor
x=   1.13 phi= -0.50 -0.05 -0.09
x=   1.25 phi= -1.07 -0.11 -0.19
x=   1.38 phi= -1.74 -0.17 -0.28
x=   1.50 phi= -2.52 -0.25 -0.38
x=   1.63 phi= -3.42 -0.33 -0.49
x=   1.75 phi= -4.46 -0.44 -0.61
x=   1.88 phi= -5.67 -0.55 -0.73
x=   2.00 phi= -7.08 -0.69 -0.86

        (may appear in a different order, each line written by a
        different processor, when it is ready)
```

3. Nonlinear Systems

The algorithm used is based on Gragg's Method (1964,1965) and polynomial extrapolation as described by Lustman, Neta and Gragg (1991). One can solve

(2)
$$y'(x) = f(x, y(x))$$
$$y(a) = y_a$$

where y and f are vector valued functions and y_a is a vector of initial values.

The host and node programs are supplied along with exa.f file containing subroutines for the evaluation of the exact solution (putex) and the right hand side (rhs) of (2). The make file to compile and link these programs is given at the end followed by an example of input and output files for the given putex and rhs.

```
c
c                HOST
c        program for the solution of nonlinear systems
c        based on Gragg's method and polynomial extrapolation
c        on INTEL iPSC/2 having 8 (maxproc) processors
c
c      see Lustman, Neta and Gragg
c
c    leny0   =   length of vector of initial values
c    nptmax  =   maximum number of points in common to all processors
c
         implicit double precision (a-h,o-z)
         parameter(leny0=20,nptmax=100)
         parameter(maxproc=8,iv=5)
         parameter(initype=1000,inilen=4*(iv+leny0)
     ,,nodes=-1,idhost=2,nodepid=3)
         dimension y0(leny0),sendata(iv+leny0)
         call getcube('extrap',' ',' ',1)
         call setpid(idhost)
         nproc=numnodes()
          print*,' got the maximal cube,',nproc,' nodes'
         call load('node',nodes,nodepid)
c
c   xmin, xmax = the interval of integration
c
         print*,'Enter xmin,xmax'
         read*,xmin,xmax
         print*,'How many result points (excluding xmin)?'
         read*,npt
         print*,'Enter dimension of solution vector'
         read*,leny
         if(leny.gt.leny0) then
         print*,'dimension=',leny,'>',leny0
         stop
         endif
         print*,'Enter ',leny,' initial values'
         read*,(y0(i),i=1,leny)
cdebugc if debugging, replace the two lines above by
cdebug   call putex(xmin,leny,y0)
           print*,'How many processors will be used?'
         read*,nn
         if(nn.gt.nproc.or.nn.lt.1) then
         print*,nn,' is unreasonable. '
         nn=nproc
         endif
         nproc=nn
         print*,' will use ',nproc,' processors'
         sendata(1)=xmin
         sendata(2)=xmax
         sendata(3)=leny
         sendata(4)=npt
         sendata(5)=nproc
         do 1 j=1,leny
   1     sendata(iv+j)=y0(j)
         call csend(initype,sendata,inilen,nodes,nodepid)
```

15

```
call waitall(nodes,nodepid)
call relcube('extrap')
stop
end
```

```
c'
c               NODE
c        program for the solution of nonlinear systems of ODEs
c        based on Gragg's method and polynomial extrapolation
c        on INTEL iPSC/2 having 8 (maxproc) processors
c
c      see Lustman, Neta and Gragg
c
       implicit double precision (a-h,o-z)
       parameter(leny0=20,nptmax=100)
       parameter(maxproc=8,iv=5)
       parameter(iii=5,jdata=iii+leny0+nptmax*leny0)
       parameter(initype=1000,inilen=4*(iv+leny0)
     ,,nodes=-1,idhost=2,nodepid=3)
       dimension y0(leny0),dataini(iv+leny0)
       dimension ysave(leny0,0:nptmax)
     ,,y(leny0),yexa(leny0),hlfway(leny0)
       dimension data(jdata)
       dimension hvec(0:maxproc)
       me=mynode()
       iam=me
       call crecv(initype,dataini,inilen)
       xmin=    dataini(1)
       xmax=    dataini(2)
       leny=    dataini(3)
       npt=     dataini(4)
       nproc=   dataini(5)
       lastproc=(nproc-1)
       if(iam.gt.lastproc) stop
       jdta=iii+leny+npt*leny
c
c ABSOLUTELY ESSENTIAL: 8 bytes per double precision item
c
       lendta=8*jdta

c
c message length in bytes
c

       ne=nproc-me
c
c save results every ne steps
c
       do 1 j=1,leny
 1     y0(j) = dataini(iv+j)
       ipow=1
c
c all the h's must be known to all the processors
c
       do 10 i=0,nproc-1
       hvec(i)=(xmax-xmin)/(npt*(nproc-i))
 10    continue
       h=hvec(me)
c
c fixes the size for integration.
```

```
c'
          jndex=0
          do 2 j=1,leny
          ysave(j,jndex)=y0(j)
  2       y(j)=y0(j)
          do 3 index=1,npt*(ne)
          x=xmin+h*(index-1)
          call odestep(h,x,y,index,hlfway,leny)
c
c advances the solution
c in this form, it is a two step method, i.e.
c         h,x,y(x)   and y(x-h/2) is what you need to obtain y(x+h)
c
          if(mod(index,ne).eq.0) then
c
c save this result, it belongs to a common point
c
          jndex=jndex+1
          do 4 j=1,leny
          ysave(j,jndex)=y(j)
  4       continue
          endif
  3       continue

          if(me.ne.lastproc) then
c
c send my saved data to lastproc (who probably is done by now)
c
          l=iii
          if(jndex.ne.npt) then
          print*,' i am ',me,' jndex=',jndex
     ,,' .ne. npt=',npt
          stop
          endif
          do 6 j=0,npt
          do 6 i=1,leny
          l=l+1
          data(l)=ysave(i,j)
  6       continue
          call csend(me,data,lendta,lastproc,nodepid)
          endif
c
c i am waiting for data to do extrapolations on
c
          level=nproc-me
c
c the new data will be sent to me-1 with superscript level
c
c
          msgtyp=(me)
          if(me.eq.lastproc) msgtyp=(me-1)
 134      continue
          call crecv(msgtyp,data,lendta)
          if(msgtyp.eq.me) then
```

18

```
c·
c just save the message in ysave
c
          l=iii
          do 69 j=0,npt
          do 69 i=1,leny
          l=l+1
          ysave(i,j)=data(l)
 69       continue
          else
c
c extrapolate incoming data and ysave
c
          it=      data(1)
          itsne=   data(2)
          itspow=  data(4)
          hish=    data(5)
c
c because the error goes in powers of h**2
c
          w=1/(    (hvec(msgtyp)/hvec(msgtyp+level))**2    -1)
          l=iii
          do 7 j=0,npt
          do 7 i=1,leny
          l=l+1
          z=data(l)
          data(l)= ysave(i,j)+w*(ysave(i,j)-data(l))
          ysave(i,j)=z
c
c This prepares extrapolated data to send and saves
c the data received to extrapolate with other message data
c

 7        continue

          call csend(msgtyp,data,lendta,me-1,nodepid)
          endif
          msgtyp=msgtyp-1
          if(msgtyp.ge.0) goto 134
          if(me.ne.0) goto 1512
c
c everything done, report results
c
          hout=(xmax-xmin)/npt
          orm=0
          er=0
          do 9 j=0,npt
          x=xmin+j*hout
cdebug    call putex(x,leny,yexa)
          print900,j,x
 900      format(i5,f10.3)
          do 8 i=1,leny
          print800,ysave(i,j)
cdebug        ,,yexa(i),abs(ysave(i,j)-yexa(i))
cdebug    orm=orm+yexa(i)**2
```

19

```
cdebug     er=er+(ysave(i,j)-yexa(i))**2
  800      format(2f10.3,1pe10.2)
  8        continue
  9        continue
cdebug     print900, -999,-999.
cdebug     orm=sqrt(orm)
cdebug     er=sqrt(er)
cdebug     reler=er/orm
cdebug     print800, orm,er,reler
  1512     continue
           end
c
c     subroutine for ode stepping using Gragg's method
c
           subroutine odestep(h,x,y0,index,hlfway,l)
c
c y0,hlfway are input and output. the step is from x=x to x=x+h
c
           implicit double precision (a-h,o-z)
           parameter(leny0=20,nptmax=100)
           dimension y0(l),hlfway(l),r(leny0)
           if(index.eq.1) then
c
c this is the first step
c
           call rhs(x,y0,l,r)
           do 61 i=1,l
  61       hlfway(i)=y0(i)+h/2*r(i)
           else
c
c the general step : hlfway is at x-h/2, y0 at x
c          they advance to x+h/2, x+h correspondingly
c
           call rhs(x,y0,l,r)
           do 661 i=1,l
  661      hlfway(i)=hlfway(i)+h*r(i)
           endif
           call rhs(x+h/2,hlfway,l,r)
           do 662 i=1,l
  662      y0(i)=y0(i)+h*r(i)

c
c Gragg formula. the errors go in powers of h**2
c
           return
           end
```

```fortran
c'
c            EXA.F
c
c    putex evaluates the exact solution
c    for this examples y(i) exact = x **i
c
       subroutine putex (x,l,y)
       implicit double precision (a-h,o-z)
       dimension y(l)
       y(1)=x
       do 1 j=2,l
       y(j)=x*y(j-1)
  1    continue
       return
       end
c
c    evaluates the right hand side for the above system
c
       subroutine rhs (x,y,l,r)
       implicit double precision (a-h,o-z)
       dimension y(l),r(l)
       x2=x*x
       div=x2*x
       do 1 i=1,l-1
       r(i)=i*y(i)*y(i+1)/div
       div=div*x
  1    continue
       r(l)=l*y(l)*y(l)/x2
       return
       end
```

```
#
#                          this is the makefile
# this file is used to compile and link the host.f, node.f
#
# the command "make all" causes compilation and linking.


all :    exa.o host node

exa.o:   exa.f

host:    host.f exa.o
         f77 -o host exa.o host.f -host


node:    node.f   exa.o
         f77 -o node exa.o node.f -node


***********************************************
              example of input file for
              the subroutines in exa.f
***********************************************

1,2
2
4
1,1,1,1
5


************************************************
              example of output file for
              the above input
************************************************
  got the maximal cube,               8 nodes
 Enter xmin,xmax
 How many result points (excluding xmin)?
 Enter dimension of solution vector
 Enter               4 initial values
 How many processors will be used?
  will use             5 processors


     0      1.000
       1.000
       1.000
       1.000
       1.000
     1      1.500
       1.500
       2.250
       3.375
       5.062
```

```
'   2       2.000
   2.000
   4.000
   8.000
  15.999
```

Acknowledgements.

References

W. B. Gragg: *Repeated extrapolation to the limit in the numerical solution of ordinary differential equations*, Ph.D. dissertation, UCLA (1964)

W. B. Gragg: *On extrapolation algorithms for ordinary initial value problems*, SIAM J. Num. Anal. 2 (1965) 384–403

C. P. Katti and B. Neta: *Solution of Linear Initial Value Problems on a Hypercube*, Technical Report NPS-53-89-001, Naval Postgraduate School, Monterey CA (1989) 7pp.

L. Lustman, B. Neta and C. P. Katti: *Solution of linear systems of ordinary differential equations on an INTEL hypercube*, submitted (1990)

L. Lustman, B. Neta and W. Gragg: *Solution of ordinary differential initial value problems on an INTEL hypercube*, Technical Report NPS-53-91-008,Naval Postgraduate School, Monterey CA (1990)

DISTRIBUTION LIST

                                   No. of Copies

Director   2
Defense Tech. Inf. Center
Cameron Station
Alexandria, VA   22314

Director of Research Admin.   1
Code 012
Naval Postgraduate School
Monterey, CA   93943

Library   2
Code 0142
Naval Postgraduate School
Monterey, CA   93943

Dept. of Mathematics   1
Code MA
Naval Postgraduate School
Monterey, CA   93943

Center for Naval Analysis   1
4401 Ford Avenue
Alexandria, VA   22302-0268

Professor Beny Neta   15
Code MA/Nd
Department of Mathematics
Naval Postgraduate School
Monterey, CA   93943

Professor Naotaka Okamoto   1
Okayama University of Science
Dept. of Applied Science
Ridai-cho 1-1, Okayama 700
Japan

Professor William Gragg   5
Code MA/Gr
Department of Mathematics
Naval Postgraduate School
Monterey, CA   93943

Professor Levi Lustman   15
Code MA/Ll
Department of Mathematics
Naval Postgraduate School
Monterey, CA   93943

Dr. C.P. Katti   1
J. Nehru University
School of Computer
        and Systems Sciences
New Delhi 110067
India

Professor Paul Nelson   1
Texas A&M University
Dept. of Nuclear Engineering
        and Mathematics
College Station, TX   77843-3133

Professor I. Michael Navon   1
Florida State University
Supercomputer Computations
     Research Institute
Tallahassee, FL   32306

Professor M.M. Chawla, Head   1
Department of Mathematics
III/III/B-1, IIT Campus
Hauz Khas, New Delhi 110016
India

Professor M. Kawahara   1
Dept. of Civil Engineering
Faculty of Science
        and Engineering
Chuo University
Kasuga 1-chome 13
Bunkyo-ku, Tokyo
Japan

Professor H. Dean Victory Jr. 1
Texas Tech University
Department of Mathematics
Lubbock, TX   79409

Professor Gordon Latta   1
Code MA/Lz
Department of Mathematics
Naval Postgraduate School
Monterey, CA   93943

Professor Arthur Schoenstadt 1
Code MA/Zh
Department of Mathematics
Naval Postgraduate School
Monterey, CA   93943

Professor H.B. Keller   1
Dept. of Applied Mathematics
California Institute of
                    Technology
Pasadena, CA   91125

INTEL Scientific Computers   1
15201 N.W. Greenbrier Pkwy.
Beaverton, OR   97006

Professor R. T. Williams   1
Code MR/Wu
Naval Postgraduate School
Department of Mathematics
Monterey, CA   93943

Professor David Gottlieb   1
Brown University
Division of Applied Mathematics
Box F
Providence, RI 02012

Mike Carron   1
Advanced Technology Staff
Code CST
Naval Oceanographic Office
Stennis Space Center, MS 39522-5001